Multi-class classification with:

- Least-squares regression
- Logistic Regression
- K-NN
- Decision trees
Discriminant Functions for $K > 2$ classes

- First idea: Use $K - 1$ classifiers, each solving a two class problem of separating point in a class $C_k$ from points not in the class.

- Known as 1 vs all or 1 vs the rest classifier

- PROBLEM: More than one good answer for green region!
Discriminant Functions for $K > 2$ classes

- Another simple idea: Introduce $K(K - 1)/2$ two-way classifiers, one for each possible pair of classes.
- Each point is classified according to majority vote amongst the disc. func.
- Known as the **1 vs 1 classifier**

**PROBLEM:** Two-way preferences need not be transitive.
We can avoid these problems by considering a single $K$-class discriminant comprising $K$ functions of the form

$$y_k(x) = w_k^T x + w_{k,0}$$

and then assigning a point $x$ to class $C_k$ if

$$\forall j \neq k \quad y_k(x) > y_j(x)$$

Note that $w_k^T$ is now a vector, not the $k$-th coordinate.

The decision boundary between class $C_j$ and class $C_k$ is given by $y_j(x) = y_k(x)$, and thus it’s a $(D - 1)$ dimensional hyperplane defined as

$$(w_k - w_j)^T x + (w_{k,0} - w_{j,0}) = 0$$

What about the binary case? Is this different?

What is the shape of the overall decision boundary?
K-Class Discriminant

- The decision regions of such a discriminant are always **singly connected** and **convex**

- In Euclidean space, an object is **convex** if for every pair of points within the object, every point on the straight line segment that joins the pair of points is also within the object

- Which object is convex?
K-Class Discriminant

- The decision regions of such a discriminant are always **singly connected** and **convex**
- Consider 2 points $x_A$ and $x_B$ that lie inside decision region $R_k$
- Any convex combination $\hat{x}$ of those points also will be in $R_k$

$$\hat{x} = \lambda x_A + (1 - \lambda)x_B$$
**Proof**

- A convex combination point, i.e., \( \lambda \in [0, 1] \)

\[
\hat{x} = \lambda x_A + (1 - \lambda)x_B
\]

- From the linearity of the classifier \( y(x) \)

\[
y_k(\hat{x}) = \lambda y_k(x_A) + (1 - \lambda)y_k(x_B)
\]

- Since \( x_A \) and \( x_B \) are in \( R_k \), it follows that
  \( y_k(x_A) > y_j(x_A), \ y_k(x_B) > y_j(x_B), \ \forall j \neq k \)

- Since \( \lambda \) and \( 1 - \lambda \) are positive, then \( \hat{x} \) is inside \( R_k \)

- Thus \( R_k \) is singly connected and convex
Multi-class Classification with Linear Regression

- From before we have:
  \[ y_k(x) = w_k^T x + w_{k,0} \]
  which can be rewritten as:
  \[ y(x) = \tilde{W}^T \tilde{x} \]
  where the \( k \)-th column of \( \tilde{W} \) is \( [w_{k,0}, w_k^T]^T \), and \( \tilde{x} \) is \( [1, x^T]^T \)

- **Training:** How can I find the weights \( \tilde{W} \) with the standard sum-of-squares regression loss?

**1-of-K encoding:**
For multi-class problems (with \( K \) classes), instead of using \( t = k \) (target has label \( k \)) we often use a **1-of-K encoding**, i.e., a vector of \( K \) target values containing a single 1 for the correct class and zeros elsewhere

**Example:** For a 4-class problem, we would write a target with class label 2 as:
\[ t = [0, 1, 0, 0]^T \]
Multi-class Classification with Linear Regression

- Sum-of-least-squares loss:

\[
\ell(\tilde{W}) = \sum_{n=1}^{N} \|\tilde{W}^T \tilde{x}^{(n)} - t^{(n)}\|^2
\]

\[
= \|\tilde{X} \tilde{W} - T\|_F^2
\]

where the \(n\)-th row of \(\tilde{X}\) is \([\tilde{x}^{(n)}]^T\), and \(n\)-th row of \(T\) is \([t^{(n)}]^T\)

- Setting derivative wrt \(\tilde{W}\) to 0, we get:

\[
\tilde{W} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T T
\]
Multi-class Logistic Regression

- Associate a set of weights with each class, then use a normalized exponential output

\[ p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)} \]

where the activations are given by

\[ z_k = \mathbf{w}_k^T \mathbf{x} \]

- The function \( \frac{\exp(z_k)}{\sum_j \exp(z_j)} \) is called a softmax function
Multi-class Logistic Regression

- The likelihood
  \[ p(T|w_1, \cdots, w_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|x^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k^{(n)}(x^{(n)})^{t_k^{(n)}} \]

  with
  \[ p(C_k|x) = y_k(x) = \frac{\exp(z_k)}{\sum_j \exp(z_j)} \]

  where \( n \)-th row of \( T \) is 1-of-\( K \) encoding of example \( n \) and
  \[ z_k = w_k^T x + w_{k0} \]

- What assumptions have I used to derive the likelihood?
- Derive the loss by computing the negative log-likelihood:

  \[ E(w_1, \cdots, w_K) = -\log p(T|w_1, \cdots, w_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log[y_k^{(n)}(x^{(n)})] \]

  This is known as the **cross-entropy** error for multiclass classification

- How do we obtain the weights?
Training Multi-class Logistic Regression

- How do we obtain the weights?

\[
E(w_1, \cdots, w_K) = - \log p(T|w_1, \cdots, w_K) = - \sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log[y_k^{(n)}(x^{(n)})]
\]

- Do gradient descent, where the derivatives are

\[
\frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k, j)y_j^{(n)} - y_j^{(n)}y_k
\]

and

\[
\frac{\partial E}{\partial z_k^{(n)}} = \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = y_k^{(n)} - t_k^{(n)}
\]

\[
\frac{\partial E}{\partial w_{k,i}} = \sum_{n=1}^{N} \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} \cdot \frac{\partial z_k^{(n)}}{\partial w_{k,i}} = \sum_{n=1}^{N} (y_k^{(n)} - t_k^{(n)}) \cdot x_i^{(n)}
\]

- The derivative is the error times the input
Softmax for 2 Classes

- Let’s write the probability of one of the classes

\[ p(C_1|x) = y_1(x) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} \]

- I can equivalently write this as

\[ p(C_1|x) = y_1(x) = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp(-(z_1 - z_2))} \]

- So the logistic is just a special case that avoids using redundant parameters

- Rather than having two separate set of weights for the two classes, combine into one

\[ z' = z_1 - z_2 = w_1^T x - w_2^T x = w^T x \]

- The over-parameterization of the softmax is because the probabilities must add to 1.
Multi-class K-NN

- Can directly handle multi class problems
Multi-class Decision Trees

- Can directly handle multi class problems
- How is this decision tree constructed?